関数型言語を用いた OS の実装手法の評価

情報科学科　村上 大樹　　指導教員：粕谷 英人

1　はじめに

OS は主な役割として、ユーザが利用するソフトウェアやプログラムがハードウェアを容易かつ効率よく利用できるようにする機能を提供する。

ここでいう “容易である”とは、OS がハードウェアを抽象化し、プログラム自身が使用するハードウェアを個別に指定する必要がないということである。例えば、コンソールに Hello World と出力したいとして、コード中に、どのキーが押されたか、画面のどの位置に出力するか、フォントやサイズはどうするかなどを記述せずとも、scanf や printf のような関数を用いるだけで実現可能である。

また “効率がよい”とは、複数のプログラムが実行中であっても有限個のハードウェアリソースを共有することができるということである。もし、あるプログラムが動作を完了させるまで複数のハードウェアリソースを独占していたら、そのほかのプログラムは何もしないで待機しなければならない時間ができてしまう。OS は、複数のプログラムに対してあたかもハードウェアリソースが無限個あるかのように見せかけ、平等にリソースを分け与える。

既存の OS では、低レベルな操作がしやすいという理由で C 言語やアセンブリが使用されているが、これらの言語には強力な静的型付けとメモリの安全性が欠けており、システム障害を引き起こす原因となる可能性がある。この問題を解決するために、関数型言語を用いて OS を開発する研究がされてきた。Hosue はその一つである。

本論文では、House の仕様について述べる。その後、House と xv6 と呼ばれる OS とを、OS の重要な要素であるプロセスとページングについて比較し、House が高い安全性を確保できているか、実用的であるかを調査する。関数型言語を使用して開発された OS は複数存在するが、その中で House は比較的規模が大きい。xv6 は、教育用に用いられており、標準的な C 言語を用いた OS としては規模が小さいいため調査がしやすいという利点がある。

2　House

House(Haskell User’s Operating System and Environment)[1] は、The Programatica Project が開発した、Haskell で記述されている OS である。Haskell は純粋関数型言語であり、同じ入力に対して必ず同じ出力をする参照透過性を持つ。型安全かつメモリ安全な Haskell を用いることで、多くのバグを防ぐことを狙いとしている。Haskell がマシンのハードウェアと直接対話するために、Haskell の IO モナドと Foreign Function Interface(FFI) 拡張機能を使用している。また、同プロジェクトが開発した P-Logic と呼ばれる拡張機能を使用して、カーネルが満たすべき仕様を定式化している。

H(ハードウェア) は、スーパバイザモードで動作するプログラムをサポートするのに適した、Haskell の IO モナドの特殊なバージョンである。仮想メモリ管理、任意のユーザバイナリの保護された実行、および低レベルの IO 操作を含む Intel IA32 アーキテクチャ上に OS を構築するために必要なハードウェア機能へのアクセスを提供する。

H は安全性を目標に設計されている。ポートを使用する I/O 操作で問題が生じる可能性があるが、それ以外で H モナドの操作が Haskell ヒープの破損を引き起こすことはない。

3　xv6

xv6[2] は、ANSI C で記述された、Sixth Edition Unix のマルチプロセッサ x86 システムへの再実装である。

xv6 はページテーブルを利用することで複数のプロセスに複数のアドレス空間を持たせ、プロセスがほかのプロセスもメモリ領域を犯すことを防ぐ。

実際の仮想・物理アドレス変換は、ページテーブルとページディレクトリの 2 レベルで行われる。ページディレクトリは、対応する物理アドレスが存在するページテーブルへのマップを行う。ページテーブルのエントリは、仮想アドレスに対応する物理アドレスが存在するページが存在するか、それが書き込み可能かどうか判別できるフラグを持っている。

4　比較

規模　House は、ハードウェアを操作するための Haskell ファイルが 15 個しかない。xv6 は OS 全体でも 12379 行しかない。

仕様の検証方法　House は、P-Logic という拡張機能でカーネルの仕様を定式化していた。xv6 は仕様を表すものはなく、開発者の記憶にとどめられるのみである。

リソース管理　House は、ページのポインタを要素に持つタプルのリストでページングを行っていた。これは 1 レベル木のページテーブルに相当する。xv6 は、ページディレクトリとページテーブルの 2 レベル木構造でページングを行っていた。

5　おわりに

本論文では、Haskell という純粋関数型言語を使用して開発された House の仕様を調査した。さらに、C 言語で実装された xv6 を比較対象として、物理アドレス管理機能について比較した。今後の課題としては、プロセス管理をはじめとする OS のその他主要な機能についても比較し、二つの OS の機能の実現方法にどのような違いがあるのかをより明確にすることが挙げられる。

参考文献

[1] Hallgren, Thomas and Jones, Mark P. and Leslie, Rebekah and Tolmach, Andrew, “OA Principled Approach to Operating System Construction in Haskell”, ICFP ’05, pp. 116–128, 2005. http://ogi.altocumulus.org/~hallgren/ICFP2005/house.pdf

[2] xv6　https://pdos.csail.mit.edu/6.828/2016/xv6/book-rev9.pdf